Claims

1. A method for gene mapping from chromosome and phenotype data, which utilizes linkage disequilibrium between genetic markers m_i, which are polymorphic nucleic acid or protein sequences or strings of single-nucleotide polymorphisms deriving from a chromosomal region, wherein

i) all marker patterns P that satisfy a pattern evaluation function $e(P)$ are searched from the data, wherein

a. the marker patterns are expressions involving the genetic markers and their alleles and zero or more of the following: individual covariates, environmental variables and auxiliary phenotypes; and

b. the pattern evaluation function $e(P)$ involves some statistical measure of the association between the marker pattern P and the phenotype being studied,

ii) each marker m_i of the data is scored by a marker score $s(m_i)$, which is a function of the set S_i defined as the set of marker patterns overlapping the marker m_i and satisfying the pattern evaluation function e as defined in step (i), and

iii) the location of the gene is predicted as a function of the scores $s(m_i)$ of all the markers m_i in the data and is based on maximizing the score if the scoring function is designed to give higher scores closer to the gene, and on minimizing the score if the scoring function is designed to give lower scores closer to the gene, as is the case for instance when the scores $s(m_i)$ are marker-wise p values.

2. A method of claim 1, wherein the chromosome data consists of either haplotypes or genotypes.

3. A method of claim 1, wherein the haplotypes and genotypes referred to in the marker patterns contain flexible regions such as gaps or disjunctions.

4. A method of claim 1, wherein the marker patterns P are searched by the following algorithm:

Input

- set U of marker patterns

- evaluation function $e(P)$ for patterns P in U
- (generalization) relation $<$ for patterns in U
- where the function e and the relation $<$ are such that if $e(P)$ is true and $P' < P$, then $e(P')$ is also true

Output

- set $S = \{P \in U \mid e(P)$ *is true*$\}$ of patterns

Method

```
1. S := {}
2. // Initialize the set of evaluated patterns:
3. E := {}
4. // Start with the most general patterns:
5. Gen := {P in U | there is no P' in U, P' != P, such that P' < P}
6. // Recursively evaluate patterns in a depth first order:
7. foreach P ∈ Gen { evaluatePatterns(P) }
8. end;

9. procedure evaluatePatterns(P) {
10.     insert P into the set E
11.     if e(P) = true then {
12. insert P into set S
13. // Find all specializations of P that have not been tested yet, and
14. // evaluate them recursively:
15. Spec := {P' in U-E | P < P', P' != P, and there is no P'' in U-E, P'' != P
16.             and P'' != P', with P < P'' < P'};
17. foreach P' in Spec { evaluatePatterns(P'); }
18.     }
19. }
```

5. A method of claim 1, wherein the marker patterns P are searched by the following algorithm:

Input

- set U of marker patterns
- evaluation function $e(P)$ for patterns P in U
- frequency threshold x

Output

- set S = {P in U | $e(P)$ *and* $ae(P)$ *is true}* of patterns, where $ae(P)$ is true if and only if the frequency of pattern P exceeds a given threshold x

Method

```
20. S := {}
21. // Initialize the set of evaluated patterns:
22. E := {}
23. // Start with the most general patterns:
24. Gen := {P in U | there is no P' in U, P' != P, such that P -> P' }
25. // Recursively evaluate patterns in a depth first order:
26. foreach P in Gen { evaluatePatterns(P) }
27. end

28. procedure evaluatePatterns(P) {
29.     insert P into the set E
30.     if ae(P) = true then {
31.       if e(P) = true then insert P into set S
32.       // Find all specializations of P that have not been tested yet, and evaluate
33.       // them recursively:
34.       Spec := {P' in U-E | P' -> P, P' != P, and there is no P'' in U-E, P'' != P
35.                   and P'' != P', with P' -> P'' and P'' -> P }
36.       foreach P' in Spec { evaluatePatterns(P') }
37. }
38. }
```

6. A method of claim 1, wherein the marker patterns P are searched by the following algorithm:

Input

- marker map $M = (m_1, \ldots, m_k)$
- phenotype vector $Y = (Y_1, \ldots, Y_n)$
- haplotype matrix H of size $n * k$
- association threshold x for chi-squared test
- maximum pattern length l
- maximum number of gaps g
- maximum gap size s

Output

- set $S = \{P \text{ in } U \mid e(P) \textit{ is true}\}$ of patterns,
- where U consists of patterns on M that consist of marker-allele assignments and that adhere to parameters l, g, and i, and
- where $e(P)$ is true if and only if chi-squared test on P using haplotype matrix H and phenotypes Y exceeds the given threshold x

Method

```
39. S := {}
40. // Number of case and control chromosomes:
41. pi_A := number of disease-associated chromosomes;
42. pi_C := number of control chromosomes;
43. pi := pi_A + pi_C
44. // A lower bound for pattern frequency:
45. lb := pi_A * pi * x / (pi_C * pi + pi_A * x)
46. // Variable for iterating over different patterns:
47. P = (p_1, ... , p_k) := ('*', ... , '*')
48. for i := 1 to k {
49. // alleles(m_i) is the set of alleles of the i:th marker
50. foreach a in alleles(m_i) {
51. p_i := a
52. // Test pattern P and all its extensions:
53. checkPatterns(P, i, i, 0, 0)
54. // Reset p_i:
55. p_i := '*'
56. }
57. }
58. end

59. // Test haplotype pattern P and all patterns that can be generated by extending P
60. // from the right:
61. procedure checkPatterns(P, start, i, nr_of_gaps, gap_length) {
62. // Output strongly associated patterns
63. if chi-squared(P, M, H, Y) >= x and p_i != '*' then insert P into set S
64. // Return if extended patterns would be too long:
65. if i = k or i+1-start > l then return
66. // Return if extended patterns can not be strongly disease-associated:
```

67. if frequency of P in disease-associated chromosomes is less than lb
68. then return;
69. // Create and test legal extensions of current pattern P (3 cases):
70. // 1. Give marker $i+1$ all possible values:
71. foreach a in alleles(m_{i+1}) {
72. $p_{i+1} := a$
73. checkPatterns (P, *start*, $i+1$, *nr_of_gaps*, 0)
74. }
75. // 2. Introduce a new gap starting at marker $i+1$:
76. if $p_i \neq$ '*' and *nr_of_gaps* $< g$ and $s \geq 1$ then {
77. $p_{i+1} :=$ '*'
78. checkPatterns (P, *start*, $i+1$, *nr_of_gaps*+1, 1)
79. }
80. // 3. Extend the current gap over marker $i+1$:
81. if $p_i =$ '*' and *gap_length* $< s$ then {
82. $p_{i+1} :=$ '*'
83. checkPatterns (P, *start*, $i+1$, *nr_of_gaps*, *gap_length*+1)
84. }
85. // Before returning, reset p_{i+1}:
86. $p_{i+1} :=$ '*'
87. return
88. }

7. A method of claim 1, wherein the marker patterns P are searched by the following algorithm:

Input

- set U of marker patterns
- evaluation function $e(P)$ for patterns P in U
- (generalization) relation $<$ for patterns in U, where the function e and the relation $<$ are such that if $e(P)$ is true and $P' < P$, then $e(P')$ is also true

Output

- set $S = \{P \text{ in } U \mid e(P) \textit{ is true}\}$ of patterns

Definitions

- function *Lgg*: $U \rightarrow 2^U$, $Lgg(P) = \{ P' \text{ in } U \mid P > P' \text{ and } P' \mathrel{!}= P \text{ and there is no } P'' \text{ in } U \text{ such that } P \mathrel{!}= P'' \mathrel{!}= P' \text{ and } P > P'' > P' \}$, the set of least general generalizations of pattern *P*.

- function *Lss*: $U \rightarrow 2^U$, $Lss(P) = \{ P' \text{ in } U \mid P < P' \text{ and } P' \mathrel{!}= P \text{ and there is no } P'' \text{ in } U \text{ such that } P \mathrel{!}= P'' \mathrel{!}= P' \text{ and } P < P'' < P' \}$, the set of least special specializations of pattern *P*.

Method

```
89. S := {}
90. Q := {}
91. // Start with the most general patterns:
92. F := {P in U | there is no P' in U, P' != P, such that P' < P};
93. while F != {} {
94.     // Evaluate the candidate patterns:
95.   foreach P in F  {
96.       if e(P) = true then insert P into set S
97.       else remove P from set F
98.   }
99.   Q := Q union F
100.          // Generate a new set of candidate patterns:
101.          C := {}
102.          foreach P in F  {
103.              C := C union { P' in U | P' in Lss(P) and for all P'' in Lgg(P'):
104.                          P'' in Q }
105.    }
106.   F := C
107. }
108.         end
```

8. A method of claim 1, wherein the marker patterns *P* are searched by the following algorithm:

Input

- set *U* of marker patterns
- evaluation function $e(P)$ for patterns *P* in *U*
- frequency threshold *x*

Output

- set S = {P in U | $e(P)$ *and* $ae(P)$ *is true*} of patterns, where $ae(P)$ is true if and only if the frequency of pattern P exceeds a given threshold x

Definitions

- function *Lgg*: $U \rightarrow 2^U$, $Lgg(P)$ = { P' in U | $P \rightarrow P'$ *and* $P' \neq P$ *and there is no* P'' in U *such that* $P \neq P'' \neq P'$ *and* $P \rightarrow P'' \rightarrow P'$}, the set of least general generalizations of pattern P.

- function *Lss*: $U \rightarrow 2^U$, $Lss(P)$ = { P' in U | $P' \rightarrow P$ *and* $P' \neq P$ *and there is no* P'' in U *such that* $P \neq P'' \neq P'$ *and* $P' \rightarrow P'' \rightarrow P$}, the set of least special specializations of pattern P.

Method

```
109. S := {}
110. Q := {}
111. // Start with the most general patterns:
112. F := {P in U | there is no P' in U, P' != P, such that P -> P' };
113. while F != {} {
114.         // Evaluate the candidate patterns:
115.         foreach P in F  {
116.             if ae(P) = true then {
117.                 if e(P) = true then insert P into set S
118.             }
119.             else remove P from set F
120.         }
121.         Q := Q union F
122.         // Generate a new set of candidate patterns:
123.         C := {}
124.         foreach P in F  {
125.             C := C union { P' in U | P' in Lss(P) and for all P'' in
    Lgg(P'):
126.                                 P'' in Q }
127.         }
128.         F := C
129. }
130. end
```

9. A method of claim 1, wherein

a) the phenotype being studied is qualitative, and

b) the pattern evaluation function *e(P)* has the form *e(P) = true if and only if* $e'(P) > x$, where $e'(P)$ is the (signed) association measure χ^2 and x is a user specified minimum value, which is chosen so that the sizes of S_i are large enough, such as 20, to give statistically sufficiently reliable estimates for the gene locus, and

c) the score $s(m_j)$ of marker m_i is the size of S_i, also called marker-wise pattern frequency of m_i and denoted by $f(m_j)$.

10. A method of claim 1, wherein

a) the pattern evaluation function *e(P)* has the form *e(P) = true if and only if* $e'(P) > x$, where $e'(P)$ is the absolute frequency of pattern P in the data and x is a user-specified value, which is chosen so that the sizes of S_i are large enough, such as 20, to give statistically sufficiently reliable estimates for the gene locus, and,

b) in order to derive the score $s(m_j)$, the p value (statistical significance) of each marker pattern P in determining the phenotype being studied is evaluated, and

c) the score $s(m_j)$ is the distance between the observed p value distribution of patterns in S_i and the uniform distribution, defined as average of $(p_i - q_i) \log (p_i / q_i)$ over all $i = 1..n$, where n is the number of haplotype patterns in S_i, p_i is the ith smallest p value in S_i, and q_i is the expectation of the ith smallest p value, if the p values were randomly drawn from the uniform distribution.

11. A method of claim 10, where the p value is computed using a linear model of form $Y = \beta_1 X_1 + \ldots + \beta_k X_k + \alpha Z + \beta_0$, where the dependent variable Y is the phenotype being studied, X_1 through X_k are covariates, such as environmental factors, and Z is a dummy variable for the occurrence of the haplotype pattern, and

the coefficients α and β_* are adjusted for best fit, and then

the significance of Z as a covariate is assessed using a t test with the null hypothesis "$\alpha = 0$".

12. A method of claim 1, wherein each score $s(m_i)$ is refined by replacing it by the marker-wise p value of the score $s(m_i)$, where the statistical significance of $s(m_i)$ is measured against the null hypotheses that there is no gene effect.

13. A method of claim 12, wherein the marker-wise p values $p(m_i)$ are determined by randomly permuting phenotypes.

14. A method of claim 1, wherein the area returned from the prediction of the gene location is contiguous or fragmented or a point.

15. A method of claim 1, wherein the location of the gene, predicted as a function of the scores $s(m_i)$ and based on maximizing or minimizing the score, is predicted to the location of the marker m_i that maximizes or minimizes the marker score $s(m_i)$.

16. A method of claim 1, wherein the location of the gene, predicted as a function of the scores $s(m_i)$ and based on maximizing or minimizing the score, is predicted to the combination of most probable intervals for containing the trait-susceptibility locus that covers at most the desired proportion t ($t \in \{0,100\%\}$) of the original region obtained by taking all such points in the studied chromosomal region whose nearest marker is within the k best scoring markers, where k is selected such that the resulting area has length at most t times the length of the studied region, and where k is maximal such value.

17. A method of claim 1, wherein the location of the gene, predicted as a function of the scores $s(m_i)$ and based on maximizing or minimizing the score, is predicted to those points in the studied chromosomal region whose nearest marker scores at least y or at most y, where y is scoring function dependent and is selected so that the probability of the gene being close to the marker is sufficiently large.

18. A method of claim 1, wherein the location of the gene, predicted as a function of the scores $s(m_i)$ and based on maximizing or minimizing the score, is determined by expert investigation of the marker scores or their visualization.

19. A method of claim 1, wherein several genes are searched for simultaneously by using marker patterns that refer to several potential gene loci at the same time.

20. A computer-readable data storage medium having computer-executable program code stored thereon operative to perform a method of any of preceding claims when executed on a computer.

21. A computer system programmed to perform the method of any of claims 1 to 19.